Pratical Machine Learning Exercise

Amazon Development Center Scotland

Applicant: Carlos Eduardo Sousa Neves
Position: Machine learning Scientist - Talent Sourcing

Introduction

Question:

This exercise **aims at predicting the value a given house will sell for**, using the UK governments's land registry data. The data was downloaded from the file *2015 FULL Price Paid Data-Single file 1995-2015.csv*. All informtion about the variable contained in the data can be found here.

Approach:

The value at which a house will sell is represented by a continuous variable. In the absence of clear classes in the data, I find regression methods to be more appropriate for this analysis. A linear regression is a simple but efficient form of regression. I will compare the performance of a linear regression with the baseline method of predicting the average value for all sales. I decided to use the R as it is specifically directed for data analysis and contains all the machine learning functions wrapped into a single package. In addition, generation of reports is also made easy by using the Integrated Development Environment RStudio.
Due to the large amount of data, I connected to the University of Aberdeen's Maxwell computer cluster and used a 32Gb worker node.

Set up environment for analysis

I started by loading all the required packages for the analysis. These include:

- *data.table*: A package to efficiently load and manage large data files
- *doParallel*: A package that allows to set the number of cores used for parallelized analysis
- *caret*: A package for **C**lassification **A**nd **RE**gression **T**raining, which aims to provide a uniform inteface for various machine learning functions
- *gridExtra*: A package that allows to automatically arrange multiple plots
- *dplyr*: A package for easy data manipulation

```
#install.packages( 'data.table' )
library( 'data.table' )

#install.packages( 'doParallel' )
library( 'doParallel' )
## Loading required package: foreach
## Loading required package: iterators
## Loading required package: parallel
```

Number of cores available: 16

```
registerDoParallel( detectCores() ) # Set number of cores to
maximum

#install.packages( 'caret' )
library( 'caret' )
## Loading required package: lattice
## Loading required package: ggplot2

#install.packages( 'gridExtra' )
library( 'gridExtra' )

#install.packages( 'dplyr' )
library( 'dplyr', warn.conflicts = F )
```

Due to the use of some randomization during the analysis, I set a random *seed*, so it can be used to consistently reproduce the analysis.

```
# Select random seed
seed <- sample( 1:1000, size = 1 )
```

Seed used for this report: 288

```
set.seed( seed )
# Remove unrequired variables from environment
rm( seed )
```

After the required packages/libraries are loaded, the data is read into R using the *fread* function. The variables of interest for this analysis are selected while the data is loaded. These variables are:

- *Price* (V2): Sale price stated on the transfer deed.
- *Date of Transfer* (V3): Date when the sale was completed, as stated on the transfer deed
- *Property Type* (V5): D = Detached, S = Semi-Detached, T = Terraced, F = Flats/Maisonettes, O = Other
- *Lease duration* (V7): Relates to the tenure: F = Freehold, L= Leasehold etc.
- *Town/City* (V12): Location of the property

```
# Create a vector with the number of the relevant variables
cols_interest <- c( 2, 3, 5, 7, 12 )
# Read file
data <- fread( 'pp-complete.csv', header = F, select =
cols_interest, data.table = F, verbose = F, showProgress = F )
# Check the data structure
head( data )
##       V2               V3 V5 V7             V12
## 1  97000 1995-06-15 00:00  D  F      SHREWSBURY
## 2  66000 1995-07-31 00:00  S  F       ALDERSHOT
## 3 117500 1995-04-13 00:00  S  F         BEDFORD
## 4  30000 1995-08-18 00:00  F  F         LINCOLN
## 5  70000 1995-05-05 00:00  D  F      SCUNTHORPE
## 6  46950 1995-05-19 00:00  T  F STOCKTON-ON-TEES
# Remove unrequired variables from environment
rm( cols_interest )
```

Data pre-processing

This analysis will focus on three variables:

- Lease duration
- Property type
- Location (inside or outside of London)

Looking at the first entries of the data, it is clear that the *Date of Transfer* (V3) is excessively descriptive for the purpose of this analysis, as I am only interested in the year of transfer. Also the *Town/City* (V12) containd too many possible values. As such I simplified these variables by maintaining only the year of the *Date of Transfer* and a logic vector of whether or not the property is in London.

```
# Check for all the possible variats of LONDON's location
unique( grep( pattern = 'LONDON', data$V12, value = T) )
## [1] "LONDON"

# Add the new simplified variables
data <- data %>%
  mutate( 'Date' = gsub( pattern = '-[0-9]{2}-[0-9]{2} [0-9]{2}:
[0-9]{2}' , replacement = '', data$V3 ) ) %>%
  mutate( 'London' = data$V12 == 'LONDON')

###
# Quick check for issues with the varibale simplification #
# Is the proportion of houses in London maintained?
mean( 'LONDON' == data$V12 )
## [1] 0.07991591
mean( data$London )
## [1] 0.07991591
# How does the data look now? (look at random entries)
data[sample(nrow(data), size = 15), c(2,6,5,7)]
##                         V3 Date            V12 London
## 2988396  1998-09-18 00:00 1998         LONDON   TRUE
## 15229693 2008-01-25 00:00 2008         LONDON   TRUE
## 4819924  1999-04-12 00:00 1999  WEST BROMWICH  FALSE
## 15456795 2008-10-16 00:00 2008       LAMPETER  FALSE
## 2842478  1997-08-18 00:00 1997           BATH  FALSE
## 1560802  1996-01-26 00:00 1996     HUNTINGDON  FALSE
## 14662945 2007-07-11 00:00 2007    MANNINGTREE  FALSE
## 6776440  2001-07-27 00:00 2001    SOUTHAMPTON  FALSE
## 7943458  2002-06-07 00:00 2002        HARWICH  FALSE
## 13457007 2006-12-19 00:00 2006        HELSTON  FALSE
## 14858046 2007-10-03 00:00 2007         LONDON   TRUE
## 11009930 2004-03-05 00:00 2004       ALFRETON  FALSE
## 19547005 2014-12-15 00:00 2014         LONDON   TRUE
## 728794   1995-07-27 00:00 1995        ROMFORD  FALSE
## 14222794 2007-06-15 00:00 2007 WOODFORD GREEN  FALSE
###

# Remove old variables
data <- data %>%
  select(-V3) %>% select(-V12)

# Rename variables
names(data)<-c( 'Price', 'Type', 'Lease', 'Date', 'London' )
head(data)
##    Price Type Lease Date London
## 1  97000    D     F 1995  FALSE
## 2  66000    S     F 1995  FALSE
## 3 117500    S     F 1995  FALSE
## 4  30000    F     F 1995  FALSE
## 5  70000    D     F 1995  FALSE
## 6  46950    T     F 1995  FALSE
```

With the data cleaned, I checked each variable for any irregularities, such as missing values.

```
# Check Property type
class( data$Type )
## [1] "character"
unique( data$Type )
## [1] "D" "S" "F" "T" "O"
```

```
# 'O' stands for other. I am unsure how heterogeneous 'other' is,
so I considered these entries as uninformative and removed them
from the analysis
idx_type <- which( data$Type == 'O' )
data <- data[-idx_type,]
unique( data$Type )
## [1] "D" "S" "F" "T"
# Any missing values?
sum( is.na( data$Type ) )
## [1] 0

# Check Lease duration
class( data$Lease )
## [1] "character"
unique( data$Lease )
## [1] "F" "L" "U"
# 'U' is not described in the varible descriptions provided. I
assume it stands for 'unknown' and so removed these entries from
the analysis as uninformative.
idx_lease <- which( data$Lease == 'U' )
data <- data[-idx_lease,]
unique( data$Lease )
## [1] "F" "L"
# Any missing values?
sum( is.na( data$Lease ) )
## [1] 0

# Check Date of transfer
class( data$Date )
## [1] "character"
unique( data$Date )
##  [1] "1995" "1996" "1997" "1998" "1999" "2000" "2001" "2002"
"2003" "2004"
## [11] "2005" "2006" "2007" "2008" "2009" "2010" "2011" "2012"
"2013" "2014"
## [21] "2015" "2016"
# The 2015 data seems to contain some information from 2016! I
removed these entries as the focus of the analysis is on the
values up to 2015
idx_date <- which( data$Date == '2016' )
data <- data[-idx_date,]
unique( data$Date )
##  [1] "1995" "1996" "1997" "1998" "1999" "2000" "2001" "2002"
"2003" "2004"
## [11] "2005" "2006" "2007" "2008" "2009" "2010" "2011" "2012"
"2013" "2014"
## [21] "2015"
# Any missing values?
sum( is.na(data$Date) )
## [1] 0

# Check Location
class( data$London )
## [1] "logical"
unique( data$London )
## [1] FALSE  TRUE
# Convert logical vector into factor for ease of analysis
data$London <- as.factor( data$London )
# Change factor level names
mean( data$London == 'TRUE' ) # proportion of properties in
London prior to conversion
```

```
## [1] 0.07984511
levels(data$London) <- c( 'out London', 'in London' )
mean( data$London == 'in London' ) # proportion of properties in
London after conversion
## [1] 0.07984511
# Any missing values?
sum(is.na(data$London))
## [1] 0

# Check Price
class( data$Price )
## [1] "character"
data$Price <- as.numeric( data$Price ) # convert into a numerical
vector
class( data$Price )
## [1] "numeric"
# Any missing values?
sum( is.na( data$Price ) )
## [1] 0

# Remove unrequired variables from environment
rm ( idx_type, idx_lease, idx_date )
```

Data splitting

To perform a consistent analysis, the data has to be split into train, validation and test sets. For this analysis, the test set is considered to be all the entries from 2015, while the remaining entries are used for model training and selection.

```
# Set Test data aside
idx_test <- which( data$Date == '2015' )
train_val <- data[-idx_test,]
test <- data [idx_test,]
###
# Quick check for issues with the test set partition
# Is the test set the expected size?
sum( data$Date == '2015' )
## [1] 934544
nrow( test )
## [1] 934544
# Is the train/valiation set the expected size?
nrow( data ) - length( idx_test )
## [1] 20030135
nrow( train_val )
## [1] 20030135
# Were the dates properly split?
unique( train_val$Date )
##  [1] "1995" "1996" "1997" "1998" "1999" "2000" "2001" "2002"
"2003" "2004"
## [11] "2005" "2006" "2007" "2008" "2009" "2010" "2011" "2012"
"2013" "2014"
unique( test$Date )
## [1] "2015"
###

# Set Validation data aside
# Create a train set that is 70% of the train/validation data
```

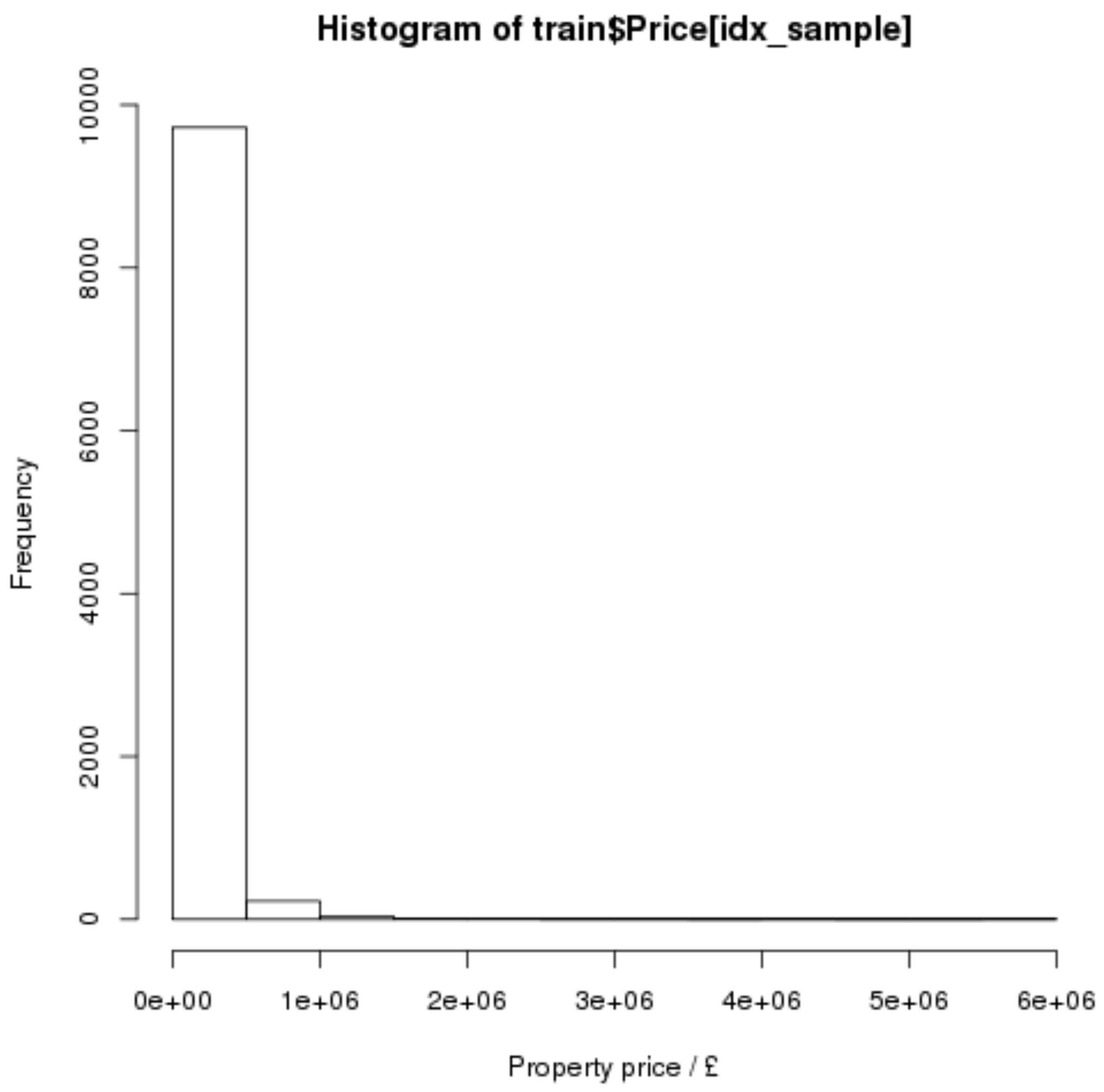
```
idx_train <- createDataPartition( train_val$Date, p = 0.7 )[[1]]
train <- train_val[idx_train,]
validation <- train_val[-idx_train,]
###
# Quick check for issues with the validation set partition
# Are the sets the expected proportions?
nrow( train ) / nrow( train_val )
## [1] 0.7000004
nrow( validation )/ nrow( train_val )
## [1] 0.2999996
# Is the data homogeneously split?
round( table( train$Date ) / length( train$Date ), digits = 3 )
##
##  1995  1996  1997  1998  1999  2000  2001  2002  2003  2004
2005  2006
## 0.040 0.048 0.055 0.052 0.060 0.056 0.062 0.067 0.063 0.063
0.053 0.066
##  2007  2008  2009  2010  2011  2012  2013  2014
## 0.063 0.032 0.031 0.033 0.033 0.033 0.040 0.048
round( table( validation$Date )/ length( validation$Date ),
digits = 3 )
##
##  1995  1996  1997  1998  1999  2000  2001  2002  2003  2004
2005  2006
## 0.040 0.048 0.055 0.052 0.060 0.056 0.062 0.067 0.063 0.063
0.053 0.066
##  2007  2008  2009  2010  2011  2012  2013  2014
## 0.063 0.032 0.031 0.033 0.033 0.033 0.040 0.048

# Remove unrequired variables from environment
rm( data, idx_test, idx_train, train_val )
# Existing variables
ls()
## [1] "test"       "train"      "validation"
```

Exploratory data analysis

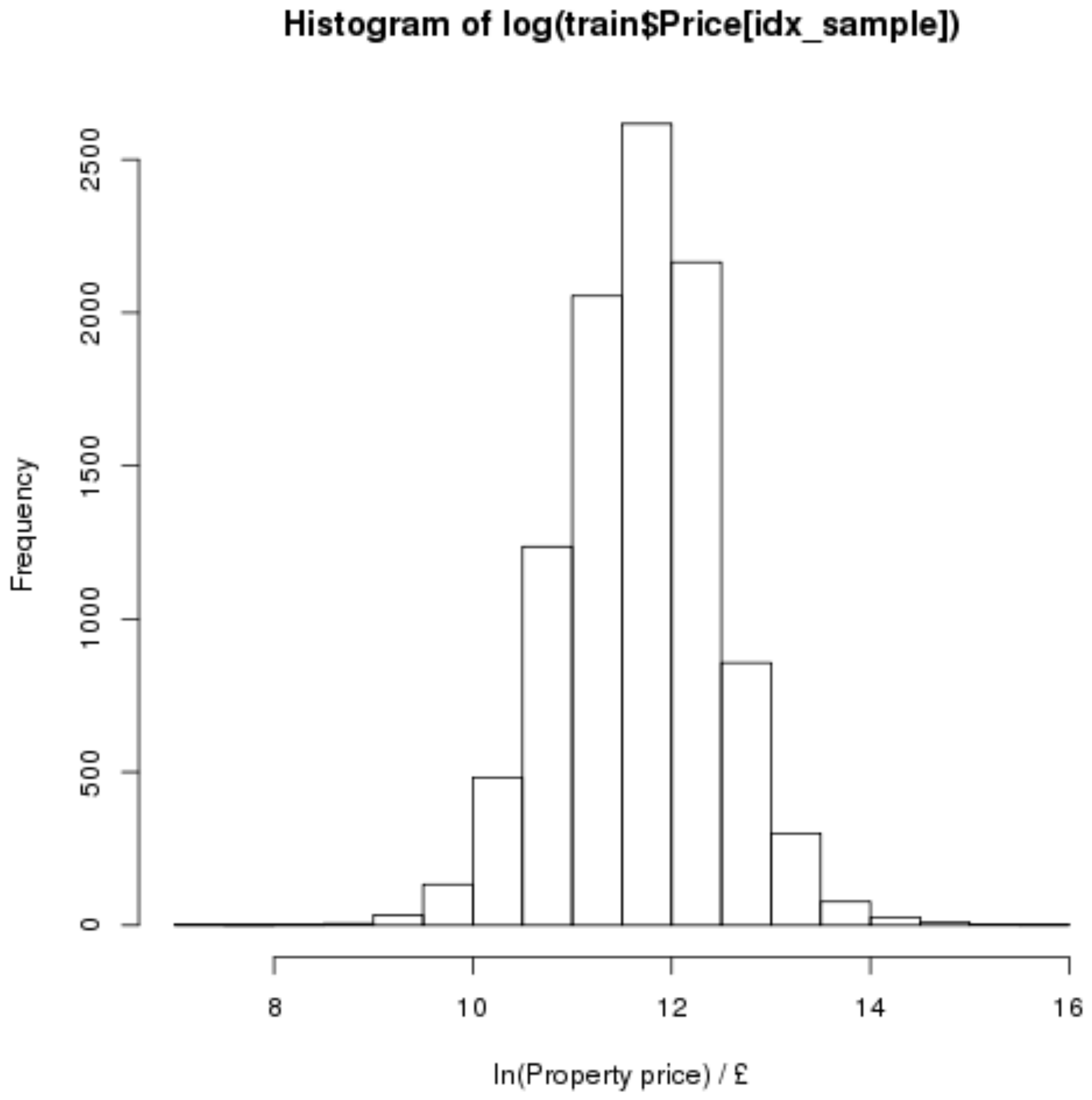
With all the data processed and split appropriately, I looked at the distribution of prices and how these relate to the predictor (categorical) variables.

```
# Take a subset sample of the data to speed the exploratory
analysis
idx_sample <- sample( nrow( train ), size = 10000 )
hist( train$Price[idx_sample], xlab = 'Property price / £' )
```



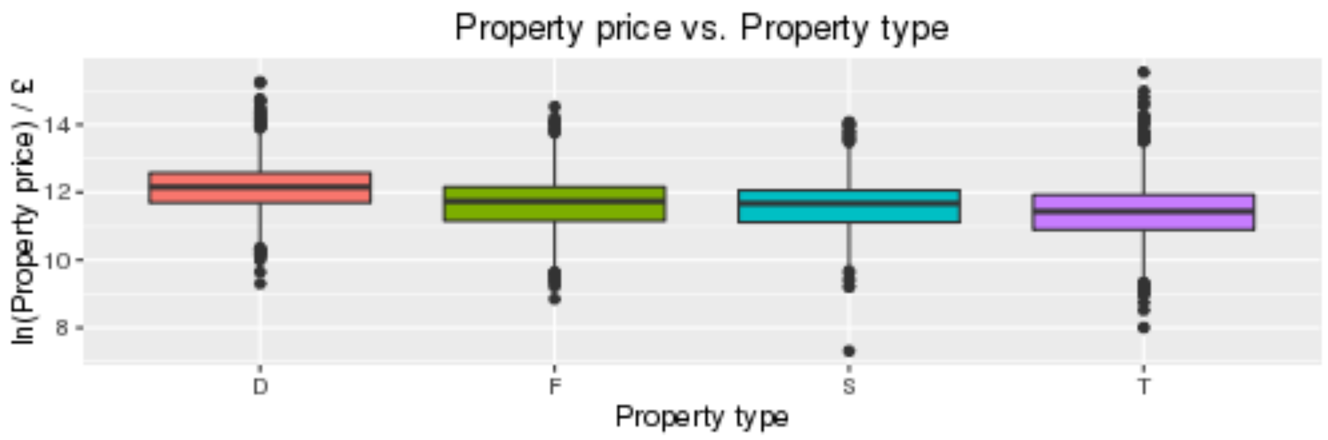
The distribution of prices is highly skewed. As this could interfere with the regression analysis, I applied the log transform to the data.

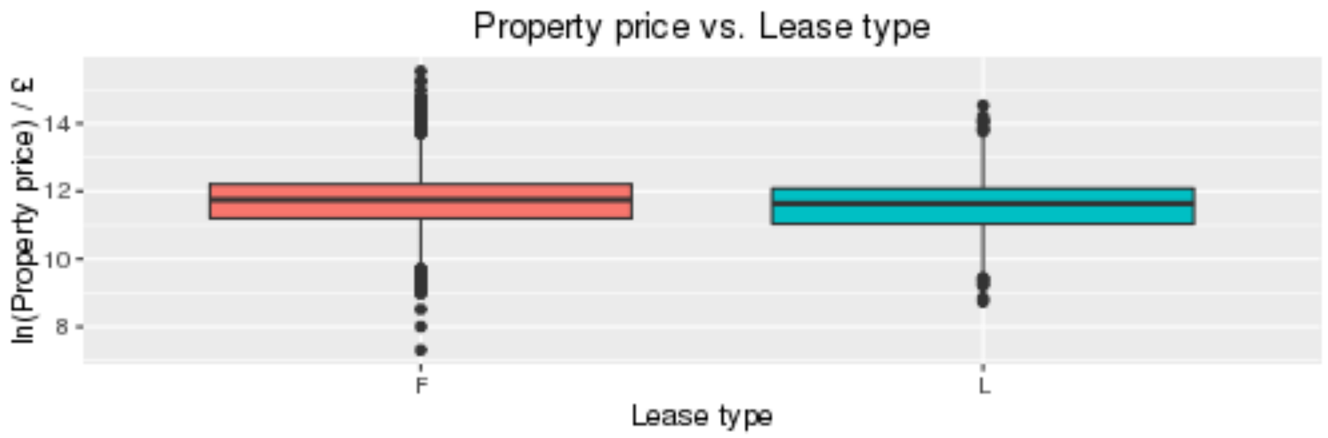
```
hist( log( train$Price[idx_sample] ), xlab = 'ln(Property price)
/ £' )
```

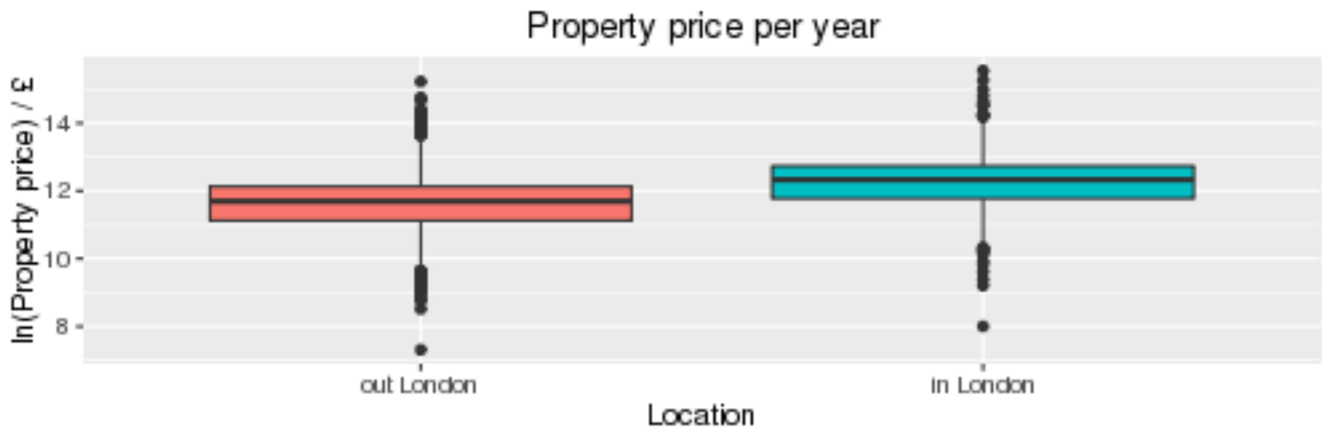


The price distribution is now much more regular (normally distributed).

```
# Plot the relationship between price and the predictor variables
# Plot Property price vs. Property type
p_type <- ggplot( train[idx_sample, ], aes( x = Type, y = log(
Price ), fill = Type ) ) + geom_boxplot() + theme(
legend.position = 'none' ) + ggtitle( 'Property price vs.
Property type' ) + ylab( 'ln(Property price) / £' )  + xlab(
'Property type' )
# Plot Property price vs. Lease  type
p_lease <- ggplot( train[idx_sample, ], aes( x = Lease, y = log(
Price ), fill = Lease ) ) + geom_boxplot() + theme(
legend.position = 'none' ) + ggtitle( 'Property price vs. Lease
type' ) + ylab( 'ln(Property price) / £' )  + xlab( 'Lease type'
)
# Plot Property price vs. Location
p_london <- ggplot( train[idx_sample, ], aes( x= London, y = log(
Price ), fill = London ) ) + geom_boxplot() + theme(
legend.position = 'none' ) + ggtitle( 'Property price per year' )
+ ylab( 'ln(Property price) / £' )  + xlab( 'Location' )
# Make image of all plots
grid.arrange( p_type, p_lease, p_london )
```







```
# Plot Property price vs. Date (for latter use)
p_date <- ggplot( train[idx_sample, ], aes( x= Date, y = log(
Price ), fill = Date ) ) + geom_boxplot() + theme(
legend.position = 'none' ) + ggtitle( 'Property price per year' )
+ ylab( 'ln(Property price) / £' )  + xlab( 'Year of transaction'
)

# Remove unrequired variables from environment
rm( idx_sample, p_type, p_london, p_lease)
```

All the predictive variables seem to have some influence on the property price. From the plots, the property type and its location (in or out of London) seem to have more varied prices than the lease type.

Model building

I decided to build a baseline model in order to understand if the regression models are in fact useful. My baseline model will assume that the prediction made for the price of the house is the most likely value. As such, it will simply predict the median price of all the houses, regardless of the house specifications. My other models will be two linear regressions either considering *Lease type* or not. To increase their accuracy, I will use k-fold cross-validation (k = 10) when training these last two models.

```
# Base model
model_base <- median( log( train$Price ) )

# Linear regression model
model_lr1 <- train( log( Price ) ~ Type + London, data = train,
method = 'lm', trControl = trainControl(method = 'cv', number =
10) )

model_lr2 <- train( log( Price ) ~ Type + London + Lease, data =
train, method = 'lm', trControl = trainControl(method = 'cv',
number = 10) )
```

Model selection

Since the outcome variable is continuous and I addressed the question using regression analysis, I used the root mean square error (RMSE) performance metric. The RMSE provides a good estimate of the total difference between the estimated and real property prices, so that the smaller the value the more accurate the prediction is expected to be. Because the model output variable was a logarithm, I reversed it to match the prices and make the RMSE more interpretable.

```
#Cross-validate base model
postResample (pred = exp( rep( model_base, times = nrow(
validation ) ) ), obs = validation$Price)[1]
##     RMSE
## 195215.3

# Cross-validate model 1
summary( model_lr1 )
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -12.4203  -0.4811   0.0483   0.4795   5.5244
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)        12.1153363  0.0003904 31035.6   <2e-16 ***
## TypeF              -0.6645273  0.0006220 -1068.3   <2e-16 ***
## TypeS              -0.5225612  0.0005272  -991.2   <2e-16 ***
## TypeT              -0.7561738  0.0005173 -1461.7   <2e-16 ***
## `Londonin London`   0.8275199  0.0007366  1123.5   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7013 on 14021098 degrees of freedom
## Multiple R-squared:  0.1847, Adjusted R-squared:  0.1847
## F-statistic: 7.942e+05 on 4 and 14021098 DF,  p-value: < 2.2e-
16
postResample (pred = exp( predict( model_lr1, newdata =
validation ) ), obs = validation$Price )[1]
##     RMSE
## 184376.5
```

```
# Cross-validate model 2
summary( model_lr2 )
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -12.5478  -0.4735   0.0473   0.4757   5.4981
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)        12.1257627  0.0003896 31122.8   <2e-16 ***
## TypeF              -0.4036117  0.0009489  -425.3   <2e-16 ***
## TypeS              -0.5119500  0.0005256  -974.1   <2e-16 ***
## TypeT              -0.7403401  0.0005167 -1432.7   <2e-16 ***
## `Londonin London`   0.8256584  0.0007332  1126.2   <2e-16 ***
## LeaseL             -0.2784899  0.0007675  -362.8   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.698 on 14021097 degrees of freedom
## Multiple R-squared:  0.1923, Adjusted R-squared:  0.1923
## F-statistic: 6.676e+05 on 5 and 14021097 DF,  p-value: < 2.2e-
16
postResample (pred = exp( predict( model_lr2, newdata =
validation ) ), obs = validation$Price)[1]
##     RMSE
## 184094.7
```

From the regression summaries, it would seem that all the predictors considered are significant for the model. In fact, both models have a root mean square error lower than the base model, with the linear regression considering the three different predictors showing a slightly better performance. Because of this, I chose this model (*model_lr2*) to predict the house values.

Model testing

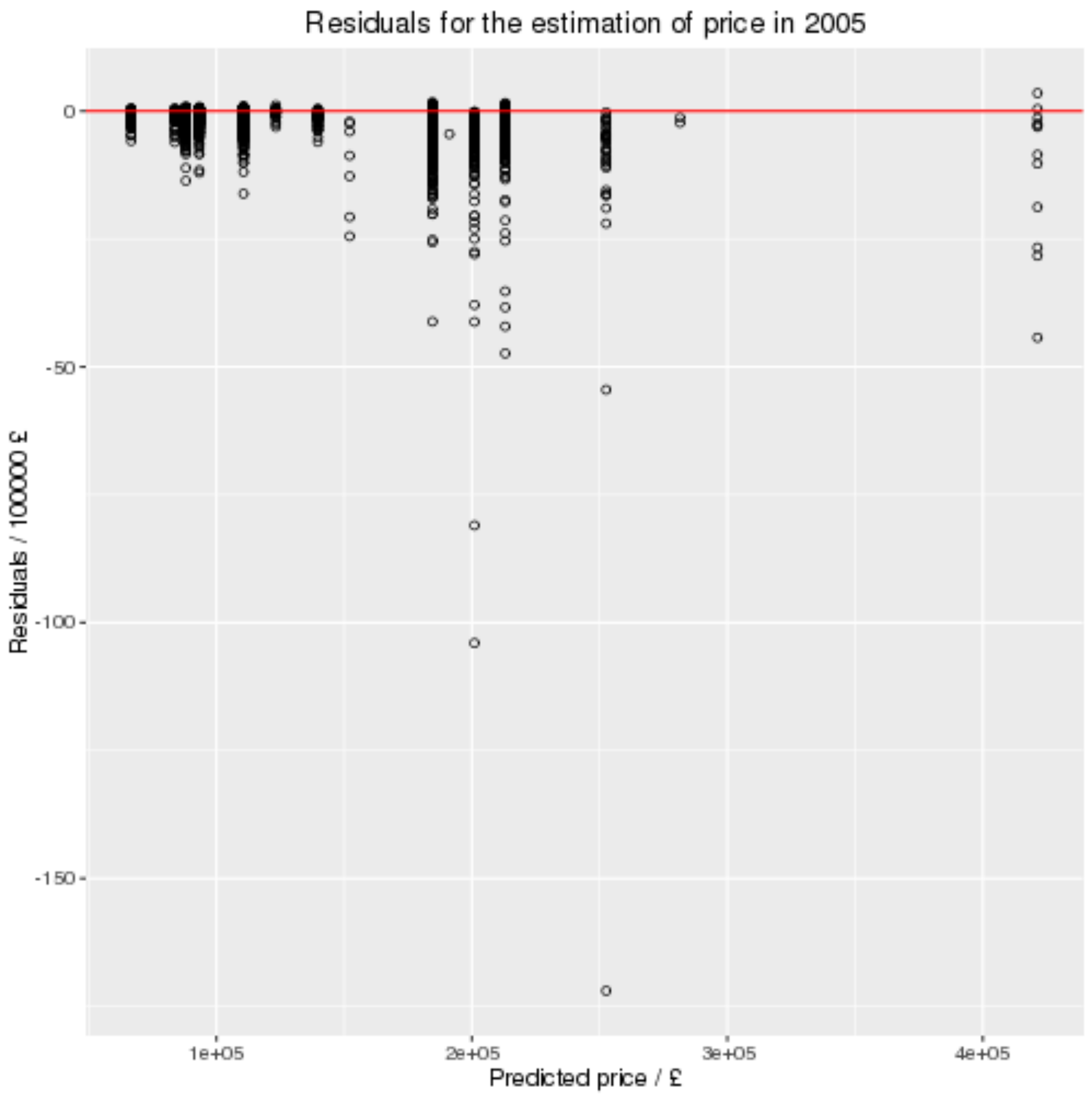
I applied the chosen model to the test set, estimated the RMSE performance measures and made a residuals plot to better understand how the model is performing.

```
# Remove unrequired variables from environment
rm( model_base, model_lr1, train, validation)

# Model performance
predicted <-  exp( predict( model_lr2, newdata = test ) )
postResample( pred = predicted, obs = test$Price )[1]
##     RMSE
## 342054.7

# Residual plot
idx_sample <- sample( length( predicted ), size = 10000 )
predicted_sample <- predicted[idx_sample]
residuals <- ( predicted_sample - test$Price[idx_sample] ) /
100000

ggplot(data.frame( residuals, predicted_sample ), aes( x =
predicted_sample, y = residuals ) ) + geom_point( shape = 1 ) +
ggtitle( 'Residuals for the estimation of price in 2005' ) +
geom_hline( yintercept = 0, color = 'red' ) + ylab( 'Residuals /
100000 £' )  + xlab( 'Predicted price / £' )
```



```
# Remove unrequired variables from environment
rm( idx_sample, predicted_sample, residuals)
```

The residuals plot shows a clear y-axis unbalance (with a few large deviations). This is most likely related to the skewed distribution of prices, where most property prices are less than £400k, but a few outlying cases cost more than £800k (*please refer to the first histogram*). Even though I did some variable transformation, this results in a systemic under-evaluation of the property price by the model.

Predicting house values

Finally, I used the model to illustrate some of its potential uses. For example, it may be relevant to consider the price difference for a similar property in London, compared to outside of London, if one is considering investing in the property market.

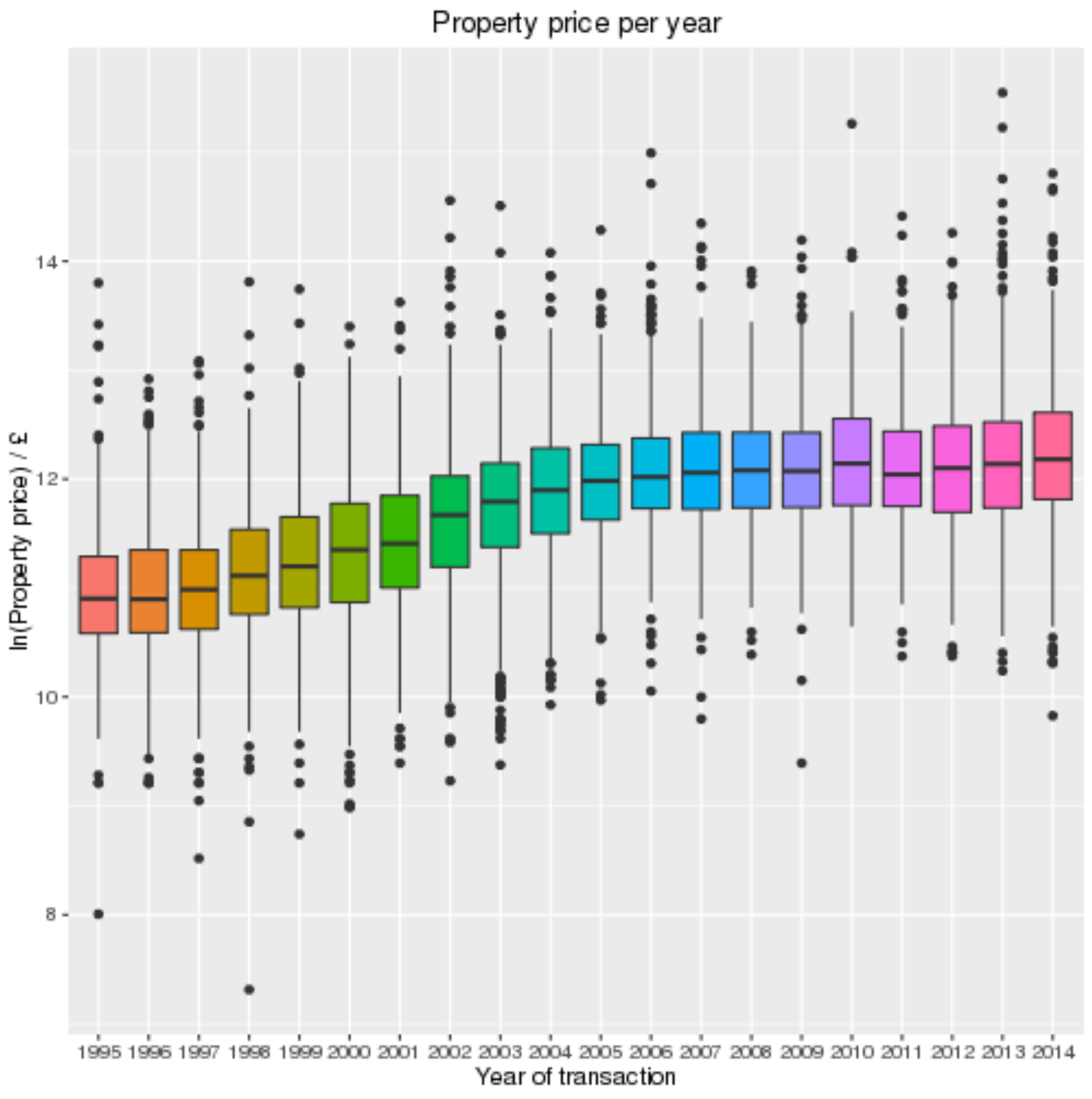
```
house1 <- c('Type' = 'D', 'Lease' = 'F', 'London' = 'out London')
house2 <- c('Type' = 'D', 'Lease' = 'F', 'London' = 'in London')
prices <- exp( predict( model_lr2, newdata = t( data.frame(
house1, house2 ) ) ) )
```

A detached house with freehold lease in London is predicted to sell for £421435.1, while a similar house outside London would sell for £184566.06. That's a difference of £236869.05!

Conclusions

The model trained in this exercise is an initial attempt to answer the question. This exercise illustrated the various steps from raw data to predictions, including data pre-processing, exploratory analysis, cross-validation, model selection and model evaluation.
In this particular case, I find the model to be biased to negative residuals (under-valuation) when tested with new data. This could potentially be solved by including additional varibles to the model, particularly the date of transaction, since all predictions were of 'future' prices (in respect to the training data). The effect of the year of sale is illustrated below:

```
p_date
```



```
# Remove unrequired variables from environment
rm( house1, house2, prices, model_lr2, p_date, predicted, test)
```